Sort Colors (Dutch National Flag Algorithm) - Revision Notes

■ Problem

Sort an array containing only 0, 1, and 2 (colors → Red, White, Blue) in-place.
Example: [2,0,2,1,1,0] → [0,0,1,1,2,2]

■ Key Idea (Three Pointers)

We divide the array into 3 regions using low, mid, high pointers:

- 0 Region → [0 … low-1] → all 0s
- 1 Region → [low … mid-1] → all 1s
- Unexplored Region → [mid … high]
- 2 Region → [high+1 … n-1] → all 2s

■ Algorithm Steps

- Initialize: low = 0, mid = 0, high = n-1
- Traverse while mid <= high:
- If nums[mid] == 0: Swap nums[low] ↔ nums[mid], move low++, mid++
- If nums[mid] == 1: Just move mid++
- If nums[mid] == 2: Swap nums[mid] ↔ nums[high], move high-- (don’t move mid yet)
- End when mid > high

■ Code (Java)

```
class Solution { public void sortColors(int[] nums) { // T.C -> O(n), S.C -> O(1) int n = nums.length;
int low = 0, mid = 0, high = n - 1; while (mid <= high) { if (nums[mid] == 0) { swap(nums, low, mid);
low++; mid++; } else if (nums[mid] == 1) { mid++; } else { swap(nums, mid, high); high--; } } } private
void swap(int[] num, int i, int j) { int temp = num[i]; num[i] = num[j]; num[j] = temp; } }
```

■ Dry Run Example

Input: [2,0,2,1,1,0]

- Step 1: mid=0, nums[mid]=2 → swap(mid, high) → [0,0,2,1,1,2]
- Step 2: mid=0, nums[mid]=0 → swap(low, mid) → [0,0,2,1,1,2], low=1, mid=1
- Step 3: mid=1, nums[mid]=0 → swap(low, mid) → [0,0,2,1,1,2], low=2, mid=2
- Step 4: mid=2, nums[mid]=2 → swap(mid, high) → [0,0,1,1,2,2], high=3
- Step 5: mid=2, nums[mid]=1 → mid++ → mid=3
- Step 6: mid=3, nums[mid]=1 → mid++ → mid=4 → loop ends
- Final: [0,0,1,1,2,2]

■ Complexity

Time Complexity: O(n) (single traversal)
Space Complexity: O(1) (in-place, only pointers used)